1)

lista = [4,7,11,4,9,5,11,7,3,5]

lista = list(set(lista))

print (lista)

2) Es más eficiente el método del vector ordenado, ya que usa menos memoria de la maquina, además las posiciones de memoria serán continuas.

3) Esta ordenado con el algoritmo de Método de ordenamiento de selección, porque los 2 números menores están de primeros, ya que, al haberse hecho 2 “pasadas” solo se han ordenado estos 2.

4)

4) def ordenamientoShell(Lista):

    contadorSublistas = len(Lista)//2

    while contadorSublistas > 0:

      for posicionInicio in range(contadorSublistas):

        Insercion(Lista,posicionInicio,contadorSublistas)

      print("Pasadas: Cuando incremento de tamaño: ",contadorSublistas,

                                   "La lista es",Lista)

      contadorSublistas = contadorSublistas // 2

def Insercion(unaLista,inicio,brecha):

    for i in range(inicio+brecha,len(unaLista),brecha):

        valorA = Lista[i]

        posicion = i

        while posicion>=brecha and Lista[posicion-brecha]>valorA:

            Lista[posicion]=Lista[posicion-brecha]

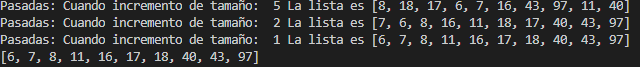
            posicion = posicion-brecha

        Lista[posicion]=valorA

Lista = [8,43,17,6,40,16,18,97,11,7]

ordenamientoShell(Lista)

print(Lista)



La pasada 1: Si se orden con un incremento 5, acercamos la lista a la ordenación obteniendo [8, 18, 17, 6, 7, 16, 43, 97, 11, 40]

La pasada 2: Luego, hacemos un incremento en 2 para obtener [7, 6, 8, 16, 11, 18, 17, 40, 43, 97]

La pasada 3: y, finalmente hacemos un incremento en 1 para ordenar la lista completamente,(véase que al ejecutar el incremento en 2, solo hace falta hacer un incremento en uno para tener la lista ordenada, ya que solo hay que cambiar de posición algunos números que están al lado de otro.)

5) lista=[1,2,1,3,2,1,2,1,3,1,2,1,2,2,2,2]

lista.sort()

m = lista[0]

a = v = lista.count(m)

for i in range (a,len(lista)):

if v < lista.count(lista[i]):

m=lista[i]

v=lista.count(m)

print("La persona con mas votos fue ", m ," Con ",v ,"votos")

break

6)A) Obtendriamos la lista de duplas ordenadas por dorsal:

[(1, 'Casillas'), (3, 'Pique'), (5, 'Puyol'), (6, 'Iniesta'), (7, 'Villa'), (8, 'Xavi Hernandez'), (11, 'Capdevila'), (14, 'Xabi Alonso'), (15, 'Ramos'), (16, 'Busquets'), (18, 'Pedrito')]

B) (key=lambda futbolista: futbolista[0]), el [0] esta indicando que se ordene de acuerdo a el primer dorsal de la dupla

C) Se obtiene lo mismo, la lista ordenada

D) [(30, 'RTX'), (70, 'Pruebas del 5g'), (99, 'IA mejorada'), (80, 'Cybertruck'), (45, 'Celulares “flip” ')]

7) def cargar():

    lista=[]

    for x in range(5):

        valor=int(input("Ingrese valor:"))

        lista.append(valor)

    return lista

def generar\_listas(lista):

    listanega=[]

    for x in range(len(lista)):

        if lista[x]<0:

            listanega.append(lista[x])

    return [listanega]

def imprimir(lista):

    for x in range(len(lista)):

        print(lista[x])

lista=cargar()

listanega=generar\_listas(lista)

print("Lista con los valores negativos")

imprimir(listanega)

En el peor de los casos se tendrá que insertar de nuevo la lista anterior, y recorrerla entera para verificar sus números negativos, su tiempo de ejecución seria casi 0.

8) [21,1] Porque la función continuara moviéndose hasta el inicio de la lista para encontrarse con el caso base.

9)

Complejidad : 3

Este algoritmo permite crear “sets” por medio de funciones, añadirle elementos, quitarle, elementos, determinar si hay o no elementos, retornar la cantidad de elementos, encuentra la posición de un elemento X en una lista ordenada, determina si es o no un subgrupo.

Def init, crea un “set” vacío

Def len, determina la longitud del “set”

Def contains, determina si X elemento pertenece o no a “set”

Def add, añade un elemento al “set”

Def remove, elimina X elemento del “set”

Def isSubsetOf, Determina si “Set” es un sub de otro “set”

Def iter, Retorna un iterador de la lista de ítems

Def findposition, Encuentra la posición de un elemento X en “set”

10)

import numpy as np

matriz = np.random.randint(size=(4, 6))

input = ("que numero desea buscar")

if input in matriz:

print(matriz)

R

11)

def QuickSort(arr):

    elements = len(arr)

    if elements < 2:

        return arr

    current\_position = 0

    for i in range(1, elements):

         if arr[i] <= arr[0]:

              current\_position += 1

              temp = arr[i]

              arr[i] = arr[current\_position]

              arr[current\_position] = temp

    temp = arr[0]

    arr[0] = arr[current\_position]

    arr[current\_position] = temp

    left = QuickSort(arr[0:current\_position])

    right = QuickSort(arr[current\_position+1:elements])

    arr = left + [arr[current\_position]] + right

    return arr

A = [91,58,19,59,4,50,14,22,11,27,16,24,60,7,58,83,26,55,34,50,21,41,78,57,8,34,11,58,29,84,98,48,6,93,14,59,89,60,14,33,52,74,1,30,8,50,37,57,30,80,81,94,89,100,88,67,3,18,57,15,73,38,59,60,73,92,30,39,43,88,42,58,15,49,88,11,71,50,77,89,74,70,4,12,27,53,18,75,38,6,82,76,54,74,2,15,65,9,24]

print("Lista A: ",A)

print("Lista A ordenada: ",QuickSort(A))

B = [7,55,58,38,44,4,62,63,110,156,134,160,94,1,4,12,186,57,181,175,177,153,146,46,156,19,196,200,57,27,75,94,161,49,69,71,42,168,49,173,197,48,162,25,199,63,28,99,194,6,161,173,130,144,76,25,91,15,146,52]

print("Lista B: ",B)

print("Lista B ordenada: ",QuickSort(B))

print("Listas: A+B: ", A + B)

print("Listas A + B ordenadas por el metodo quicksort: ",QuickSort(A)+QuickSort(B))

Obteniendo esto

Lista A: [91, 58, 19, 59, 4, 50, 14, 22, 11, 27, 16, 24, 60, 7, 58, 83, 26, 55, 34, 50, 21, 41, 78, 57, 8, 34, 11, 58, 29, 84, 98, 48, 6, 93, 14, 59, 89, 60, 14, 33, 52, 74, 1, 30, 8, 50, 37, 57, 30, 80, 81, 94, 89, 100, 88, 67, 3, 18, 57, 15, 73, 38, 59, 60, 73, 92, 30, 39, 43, 88, 42, 58, 15, 49, 88, 11, 71, 50,

77, 89, 74, 70, 4, 12, 27, 53, 18, 75, 38, 6, 82, 76, 54, 74, 2, 15, 65, 9, 24]

Lista A ordenada: [1, 2, 3, 4, 4, 6, 6, 7, 8, 8, 9, 11, 11, 11, 12, 14, 14, 14, 15, 15, 15, 16, 18, 18, 19, 21, 22, 24, 24, 26, 27, 27, 29, 30, 30, 30, 33, 34, 34, 37, 38, 38, 39, 41, 42, 43, 48, 49, 50, 50, 50, 50, 52, 53, 54, 55, 57, 57, 57, 58, 58, 58, 58, 59, 59, 59, 60, 60, 60, 65, 67, 70, 71, 73, 73, 74, 74,

74, 75, 76, 77, 78, 80, 81, 82, 83, 84, 88, 88, 88, 89, 89, 89, 91, 92, 93, 94, 98, 100]

Lista B: [7, 55, 58, 38, 44, 4, 62, 63, 110, 156, 134, 160, 94, 1, 4, 12, 186, 57, 181, 175, 177, 153, 146, 46, 156, 19, 196, 200, 57, 27, 75, 94, 161, 49, 69, 71, 42, 168, 49, 173, 197, 48, 162, 25, 199, 63, 28, 99, 194, 6, 161, 173, 130, 144, 76, 25, 91, 15, 146, 52]

Lista B ordenada: [1, 4, 4, 6, 7, 12, 15, 19, 25, 25, 27, 28, 38, 42, 44, 46, 48, 49, 49, 52, 55, 57, 57, 58, 62, 63, 63, 69, 71, 75, 76, 91, 94, 94, 99, 110, 130, 134, 144, 146, 146, 153, 156, 156, 160, 161, 161, 162, 168, 173, 173, 175, 177, 181, 186, 194, 196, 197, 199, 200]

Listas: A+B: [24, 58, 19, 59, 4, 50, 14, 22, 11, 27, 16, 24, 60, 7, 58, 83, 26, 55, 34, 50, 21, 41, 78, 57, 8, 34, 11, 58, 29, 84, 48, 6, 14, 59, 89, 60, 14, 33, 52, 74, 1, 30, 8, 50, 37, 57, 30, 80, 81, 89, 88, 67, 3, 18, 57, 15, 73, 38, 59, 60, 73, 30, 39, 43, 88, 42, 58, 15, 49, 88, 11, 71, 50, 77, 89, 74, 70, 4, 12, 27, 53, 18, 75, 38, 6, 82, 76, 54, 74, 2, 15, 65, 9, 91, 93, 92, 100, 98, 94, 6, 4, 1, 4, 7, 55, 62, 63, 110, 156, 134, 160, 94, 58, 38, 12, 186, 57, 181, 175, 177, 153, 146, 46, 156, 19, 196, 200, 57, 27, 75, 94, 161, 49, 69, 71, 42, 168, 49, 173, 197, 48, 162, 25, 199, 63, 28, 99, 194, 44, 161, 173, 130, 144, 76, 25, 91, 15, 146, 52]

Listas A + B ordenadas por el metodo quicksort: [1, 2, 3, 4, 4, 6, 6, 7, 8, 8, 9, 11, 11, 11, 12, 14, 14, 14, 15, 15, 15, 16, 18, 18, 19, 21, 22, 24, 24, 26, 27, 27, 29, 30, 30, 30, 33, 34, 34, 37, 38, 38, 39, 41, 42, 43, 48, 49, 50, 50, 50, 50, 52, 53, 54, 55, 57, 57, 57, 58, 58, 58, 58, 59, 59, 59, 60, 60, 60, 65, 67, 70, 71, 73, 73, 74, 74, 74, 75, 76, 77, 78, 80, 81, 82, 83, 84, 88, 88, 88, 89, 89, 89, 91, 92, 93, 94, 98, 100, 1, 4, 4, 6, 7, 12, 15, 19, 25, 25, 27, 28, 38, 42, 44, 46, 48, 49, 49, 52, 55, 57, 57, 58, 62, 63, 63, 69, 71, 75, 76, 91, 94, 94, 99, 110, 130, 134, 144, 146, 146, 153, 156, 156, 160, 161, 161, 162, 168, 173, 173, 175, 177, 181, 186, 194, 196, 197, 199, 200]

12) Radixsort: Es un algoritmo de ordenación que ordena tomando en cuenta cada digito, usado para números decimales, y fechas sobre todo.

Binsort: Crea un array con “baldes vacíos”, vuelve a su array y pone cada objeto en un balde, luego busca el balde vacío busca los baldes en orden y pone los elementos otra vez en el array